IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of

Chen et al.

Serial No.: 10/808,429 **Group Art Unit:** 2166

Filed: March 25, 2004 **Examiner:** Johnese T. Johnson

For: SYSTEM AND METHOD FOR BUSINESS OBJECT DISCOVERY

Honorable Commissioner of Patents
Alexandria, VA 22313 - 1450

APPELLANT'S PRE-APPEAL BRIEF REQUEST FOR REVIEW

Sir:

Comes now the Appellants and respectfully request that the Pre-Appeal Brief Conference Panel withdraw the rejections by the Examiners of claims 1 and 3-41 in the Office Action dated October 6, 2010. This request is filed concurrently with the filing of a Notice of Appeal.

I. THE CLAIMED INVENTION

An exemplary embodiment of the claimed invention, as recited by, for example, independent claim 1, is directed to a method of discovering a business object definition that includes receiving an object and a collaboration code, and determining a business object definition for the object *based upon the collaboration code*. (Page 11, lines 14-20). The collaboration code determines the business object definition for the object without pre-defined business object definitions, if the object does not conform to a known business object definition. (Page 13, line 8-page 14, line 7).

II. THE ALLEGED PRIOR ART REFERENCES

A. Mitchell and Kaipa

The Examiner alleges that Mitchell, when combined with Kaipa, teaches or suggests all of the claimed features as recited by claims 1 and 3-41. Applicant submits, however, that these references would not have been combined and even if combined, the combination would not teach or suggest each and every element of the claimed invention.

Claim 1 recites, inter alia: "*receiving an object and a collaboration code; determining a business object definition for said object based upon said collaboration code; and storing said business object definition, wherein said collaboration code determines said business object definition for said object without pre-defined business object definitions, if the object does not conform to a known business object definition.*"

Support for the claimed features of claim 1 may be found in at least page 10, line 15-page 14, line 8 of the Specification.

Claims 13, 20, 25, and 33 recite similar features as those recited by claim 1.

Instead, the Examiner merely repeated her rejections of the claims verbatim, as stated in paragraph 4 of the Office Action of April 14, 2010 (Office Action, page 2, paragraph 3 and paragraph 6(Response to arguments)), without explaining how the cited sections of the references correspond to each feature of the claimed invention, and without addressing any of the issues raised by the Applicants in the Amendment of July 9, 2010 regarding the deficiencies of Mitchell.

That is, the Examiner relies on paragraph 33 and 35 of Mitchell to attempt equating a procedure to manipulate data/ object as the claimed "*collaboration code.*" (Office Action, page 2, paragraph 4). However, paragraph 33 merely teaches that in order to aggregate new behavior and functionality to an interface or running object, the object must first be imported into the middleware network. To do this, it is necessary to gain access to the meta types for runtime objects. Also, paragraph 35 merely teaches that a illustrated implementation of the middleware framework described above supports automated importation of SOAP (Simple Object Access Protocol) interfaces. The XML meta data for other runtime types (for example, C++) may be handcrafted or generated with other methods. The interface recognizer that allows automated importation of SOAP interfaces (using a WSDL parser) may be modified to automate importation of COM, CORBA, RPC, Java, or any other type with runtime interfaces that are either discoverable or reverse engineerable. Entire libraries (in the context of adding types to a library, for example) may also be imported in this manner. Both paragraphs do not teach or suggest a procedure for manipulate data/object, and do not teach or suggest the claimed "*collaboration code.*" The Examiner also has not identified which part of the paragraphs 33 or 35 corresponds with the alleged procedure, or teaches or suggests "*collaboration code.*" Therefore, the Examiner has not met her burden to prove that Mitchell teaches or suggests, "*receiving an object and a collaboration code;*"

The Examiner also relies on paragraph 34, lines 4-8 and paragraph 38, lines 3-5, and again attempts to equate an alleged procedure(s) to manipulate the data/object as the claimed "*collaboration code*" to allege that Mitchell teaches or suggests, *determining a business object definition for said object based upon said collaboration code;*" However, as explained previously, Mitchell fails to teach or suggest, and the Examiner has failed to meet his burden to prove that Mitchell teaches or suggests, "*collaboration code.*" Also, paragraph 34 of Mitchell merely teaches that the reason the imported object remains unaffected by the aggregation is that the middleware system automatically creates a Meta Data Definition Object (MDDO) from the discoverable type definitions of the imported object. Here, the Examiner appears to cite Mitchell's MDDO and discoverable type definition out-of-context, **without understanding** that Mitchell's MDDO definition is based on the imported object, rather than "*collaboration code.*" Similarly, paragraph 38 merely teaches that when a type is recognized by the Interface Recognizer 107, the middleware automatically generates an MDDO (Meta Data Definition Object). The MDDO may be a meta object instance of the class structure contained in the meta interfaces of the imported type. Here, the Examiner again mistaken MDDO as the claimed "*business object definition,*" and has not identified which structure of Mitchell teaches or suggests, "*collaboration code*" that the "*business object definition*" is based on. Therefore, the Examiner has failed to meet her burden to prove that Mitchell teaches or suggests, *determining a business object definition for said object based upon said collaboration code;*"

Kaipa also fails to remedy Mitchell's deficiencies.

The Examiner has not even alleged, and Kaipa fails to teach or suggest, *"receiving an object and a collaboration code"* and *"determining a business object definition for said object based upon said collaboration code."* Instead, the Examiner merely alleges that Kaipa teaches or suggests, *storing said business object definition, wherein said collaboration code determines said business object definition for said object without pre-defined business object definitions, if the object does not conform to a known business object definition."* (Office Action, page 2, paragraph 4). However, Kaipa also does not teach or suggest *"wherein said collaboration code determines said business object definition for said object without pre-defined business object definitions, if the object does not conform to a known business object definition."*

The Examiner relies on paragraph 39, lines 1-3 and 7-10 of Kaipa to allege that Kaipa teaches or suggests, *"wherein said collaboration code determines said business object definition for said object without pre-defined business object definitions, if the object does not conform to a known business object definition."*

(Office Action, page 3, lines 1-5). However, paragraph 39, lines 1-3 merely teaches that once loaded, the automated object discovery agent (ODA)/conversion agent 311 proceeds to apply the mapping and naming criteria and generate a business object definition, without teaching or suggesting that Kaipa's business object definition is determined "*without pre-defined business object definitions,*" under the condition, "*if the object does not conform to a known business object definition.*" Also, paragraph 39, lines 7-10 merely teaches that the resultant business object definition (in this case, preferably a Java object) is forwarded to the connector 310 for use in runtime object. If another schema is ready to be operated on, the process is repeated.

The Examiner alleges that "(Kaipa's) definition stored in a business object is interpreted as unknown because the schema is XML and the business object definition created is java." (Office Action, page 3, lines 2-5). However, it is unclear what the Examiner meant. Here, it appears that the Examiner does not even have an understanding of what Kaipa teaches or suggests to one of ordinary skill in the art. Here, Kaipa does not teach or suggest, and the Examiner has not identified any element of Kaipa that corresponds with the claimed "*collaboration code*" that "*determines said business object definition for said object without pre-defined business object definitions,*" under the condition, "*if the object does not conform to a known business object definition.*" While Kaipa teaches that the object discovery agent (ODA)/conversion engine 311 proceeds to apply the mapping an naming criteria, and generate a business object definition (step 440), Kaipa fails to teach or suggest that the generation is done "*without pre-defined business object definitions,*" under the condition *"if the object does not conform to a known business object definition."* That is, Kaipa does not even make a determination, *"if the object does not conform to a known business object definition."* Whether Kaipa's business object definition is in an XML schema component or a JAVA component conversion has nothing to do with whether Kaipa's object "*conform to a known business object definition.*"

It also would not be obvious to combine Mitchell with Kaipa, even if Mitchell and Kaipa teaches or suggests all of the alleged respective features of claim 1.

In re Kahn states that, "Rejection based on obviousness grounds cannot be sustained by mere conclusory statements; instead, there must be some articulated reasoning with some rational underpinning to support the legal conclusion of obviousness. *In re Kahn,* 441 F. 3d 977, 988 The Supreme Court in *KSR International Co. v. Teleflex Inc.*, 550 U.S. ___, ___, 82 USPQ2d 1385, 1395-97 (2007) identified a number of rationales to support a conclusion of obviousness which are consistent with the proper "functional approach" to the determination of obviousness as laid down in *Graham.* The key to supporting any rejection under 35 U.S.C. 103 is the clear articulation of the reason(s) why the claimed invention would have been obvious. The Supreme Court in *KSR* noted that the analysis supporting a rejection under 35 U.S.C. 103 should be made explicit.

Exemplary rationales that may support a conclusion of obviousness include:

(A)Combining prior art elements according to known methods to yield predictable results;
(B)Simple substitution of one known element for another to obtain predictable results;
(C)Use of known technique to improve similar devices (methods, or products) in the same way;
(D)Applying a known technique to a known device (method, or product) ready for improvement to yield predictable results;
(E)"Obvious to try" – choosing from a finite number of identified, predictable solutions, with a reasonable expectation of success;
(F)Known work in one field of endeavor may prompt variations of it for use in either the same field or a different one based on design incentives or other market forces if the variations are predictable to one of ordinary skill in the art;
(G)Some teaching, suggestion, or motivation in the prior art that would have led one of ordinary skill to modify the prior art reference or to combine prior art reference teachings to arrive at the claimed invention. (See MPEP 2143).

Here, the Examiner merely repeated her conclusory rationale that it would have been obvious, at the time the invention was made, to have modified the teaching of Mitchell by the teaching of Kaipa to provide a better way to carry out the conversion of XML schema to Java and other object definitions. (Office Action, page 3, lines 7-10). However, the Examiner has not even provided any evidence that Kaipa's teaching is indeed a better way than Mitchell to carry out the conversion of XML schema. Indeed, Mitchell is not even concerned with converting XML schema to Java. Rather, Mitchell is concerned with dynamically aggregating data and functionality runtime objects in order to increase component reuse and ease integration of disparate objects. (See Abstract), which is unrelated to Kaipa's method for mapping and labeling XML schema elements and types (to Qualified Java components) (See Abstract and title). In fact, Mitchell does not even recognize that there is a problem with converting XML schema to Java. Similarly, Kaipa also does not improve upon Mitchell's attempt to dynamically aggregate data and functionality to runtime object in order to increase component reuse and case integration of disparate objects. Since Mitchell and Kaipa are unrelated, one of ordinary skill in the art would not have even considered combining the two references to teach or suggest all of the features of the claimed invention.

Regarding claims 3-12, 14-24, 26-32, and 34-43, the Examiner still has not provided a rationale to combine Mitchell with Kaipa to teach or suggest the dependant claims. Therefore, the Examiner has not satisfied her burden to prove that it would have been obvious to one of ordinary skill in the art at the time the invention was made to have considered combining Mitchell and Kaipa to teach or suggest the claimed inventions of the above claims.

The Examiner also attempts to combine Lai and Kaipa to teach or suggest the features of **claim 20**. (Office Action, page 13-15). However, the Examiner merely alleges that Lai teaches or suggests, "*a broker comprising a collaboration that receives a first business object and a first business object definition, that generates a second business object without a predetermined business object definition, and that generates a collaboration code, and a processor serving to execute a reverse object discovery agent that receives the first business object definition, the second business object, and the collaboration code from the broker and that defines a newly discovered business object definition for the second business object as a second business object definition during runtime as a based upon said collaboration code,*"..., and "*wherein the reverse object discovery agent determines a structure of the second business object definition by converting a structure of the second business object directly, by examining an instruction that creates the second business object,*" by citing various unrelated paragraphs of Lai, without identifying the particular structures that corresponds to each claimed feature, as required under MPEP.

Kaipa also fails to remedy Lai's deficiencies. The Examiner does not even allege, and Kaipa fails to teach or suggest, the above-recited features of claim 20. Instead, the Examiner merely alleges that Kaipa teaches, "*wherein said reverse object discovery agent determines said second business object definition for said second business object without pre-defined business object definitions, if the reverse object discovery agent determines that the second business object does not conform to a known business object definition*." (Office Action, page 14). However, for the same reasons already discussed regarding claim 1, Kaipa also fails to teach or suggest the above-recited features of claim 20 as well.

Further, the Examiner erred by making the exact same conclusory rationale to combine Lai with Kaipa, as with other references, without showing any evidence that Kaipa does indeed provide a better way to carry out the conversion of XML schema to Java and other definition than Lai, or that Lai is even concerned with the conversion of XML schema to Java at all.

The Examiner also alleges that Mitchell teaches or suggests claim 39's features "*wherein the processor for receiving the object and the collaboration code, and for determining the object definition for said object*

based on said collaboration code, and the collaboration code for determining whether the object conforms to the known business object definition, comprise a reverse object discovery agent means," (Office Action, page 7, lines 23-28) by relying on paragraph 22 of Mitchell. However, Mitchell merely teaches that Aggregation in embodiments of the present invention may depend on: the existence of an object with meta data that may be discovered or reverse-engineered, (Paragraph 22) which does not necessarily mean that there is indeed a processor that includes all of the functions as claimed, and includes all of the structural features included in the specification that describes "*a reverse object discovery agent means*." Kaipa also does not remedy Mitchell's deficiencies because the Examiner does not even allege, and Kaipa fails to teach or suggest, the above-recited features of claim 39. Therefore, the Examiner failed to satisfy her burden to prove that Mitchell and Kaipa teaches or suggests all of the features of claim 39.

With regard to claim 41, since Mitchell and Kaipa do not teach or suggest all of the features of claim 1, the references also do not teach or suggest the sequence of the steps of claim 41.

III. CONCLUSION

In view of the foregoing, Appellant respectfully submits that the rejection of claims 1-41 is clearly not proper and without basis and, therefore, requests that the Pre-Appeal Brief Conference Panel withdraw the rejection.

The Commissioner is hereby authorized to charge any deficiency in fees or to credit any overpayment in fees to Assignee's Deposit Account No. 50-0510.

Respectfully submitted,

Jeoyuh Lin, Esq.
Registration No. 56,032
Sean M. McGinn, Esq.
Registration No.: 34,386

Date: December 6, 2010

McGinn IP Law Group, PLLC
Intellectual Property Law
8321 Old Courthouse Road, Suite 200
Vienna, Virginia 22182-3817
(703) 761-4100
Customer No. 48150